Post-Quantum Zero Knowledge, Revisited
(or: How to do Quantum Rewinding Undetectably)

Alex Lombardi
(Simons & Berkeley)

Fermi Ma
(Simons & Berkeley)

Nicholas Spooner
(Warwick)
Big Question:

When are *classical* cryptosystems secure against *quantum* attacks?
(i.e., post-quantum cryptography)
Why aren’t post-quantum assumptions enough?

**Misconception:** just need to “replace” quantum-broken assumptions (factoring) with post-quantum assumptions (learning with errors).
Why aren’t post-quantum assumptions enough?

**Misconception:** just need to “replace” quantum-broken assumptions (factoring) with post-quantum assumptions (learning with errors).

\[
\text{security} = \text{definition} + \text{reduction} + \text{assumption}
\]
Why aren’t post-quantum assumptions enough?

**Misconception:** just need to “replace” quantum-broken assumptions (factoring) with post-quantum assumptions (learning with errors).

$$\text{post-quantum security} = \text{post-quantum definition} + \text{post-quantum reduction} + \text{post-quantum assumption}$$

In reality, we also need **definitions** and **reductions** that capture post-quantum security. This can be surprisingly difficult!
This work: zero-knowledge protocols [GMR85]

\[ x, w \quad \rightarrow \quad x \quad \downarrow \quad \text{accept/reject} \]
This work: zero-knowledge protocols [GMR85]

- Zero knowledge: proof (of a true statement) can be efficiently simulated without the witness.
This work: zero-knowledge protocols [GMR85]

- **Zero knowledge**: proof (of a true statement) can be efficiently simulated without the witness.
- **Completeness**: If $x$ is true and $P$ is honest, $V$ accepts.
This work: zero-knowledge protocols [GMR85]

- **Zero knowledge**: proof (of a true statement) can be *efficiently simulated* without the witness.
- **Completeness**: If $x$ is true and $P$ is honest, $V$ accepts.
- **Soundness**: If $x$ is false, $P^*$ cannot make $V$ accept.
We have a deep theory of ZK in the classical setting.

[GM85, B86, GMW86, GK96, FS90, ...]
We have a deep theory of ZK in the classical setting.

[GMR85,B86,GMW86,GK96,FS90,...]

Will this theory hold up against quantum attacks?
We have a deep theory of ZK in the classical setting. [GMR85, B86, GMW86, GK96, FS90,...]

Will this theory hold up against quantum attacks?

Positive results exist [W07, CCY20, BS20], but a cohesive theory is elusive.
We have a deep theory of ZK in the classical setting.

[GMR85,B86,GMW86,GK96,FS90,…]

Will this theory hold up against quantum attacks?

Positive results exist [W07,CCY20,BS20], but a cohesive theory is elusive. Even many *textbook ZK protocols* are still not understood!

- [Goldreich-Micali-Wigderson86]: graph non-isomorphism
- [Feige-Shamir90]: ZK via “trapdoor extraction”
- [Goldreich-Kahan96]*: five-message proofs for NP

*[CCY21] proved that [GK96] is post-quantum $\varepsilon$-ZK (a weakening of ZK).
This Work

(1) Revisit definition of post-quantum ZK.
This Work

(1) Revisit definition of post-quantum ZK.

• Overcomes a subtle definitional issue highlighted by [CCLY21].
(1) Revisit definition of post-quantum ZK.
   • Overcomes a subtle definitional issue highlighted by [CCLY21].

(2) Extract witness from quantum attacker without disturbing its state.
This Work

(1) Revisit definition of post-quantum ZK.
- Overcomes a subtle definitional issue highlighted by [CCLY21].

(2) Extract witness from quantum attacker without disturbing its state.
Consequences:
- [GMW86] GNI protocol is post-quantum ZK
- [FS90] protocol is post-quantum ZK (*super-poly assumption)
This Work

(1) Revisit definition of post-quantum ZK.
   - Overcomes a subtle definitional issue highlighted by [CCLY21].

(2) Extract witness from quantum attacker without disturbing its state.
Consequences:
   - [GMW86] GNI protocol is post-quantum ZK
   - [FS90] protocol is post-quantum ZK (*super-poly assumption)

See paper: [GK96] protocol for NP is post-quantum ZK.
Plan for Today

1. Recap: GNI protocol + “extract-and-simulate”
2. Challenges in the post-quantum setting
3. This work: defining post-quantum ZK
4. This work: quantum extract-and-simulate
Plan for Today

1. Recap: GNI protocol + “extract-and-simulate”

2. Challenges in the post-quantum setting

3. This work: defining post-quantum ZK

4. This work: quantum extract-and-simulate
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: \( G_0 = \begin{array}{c}
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\end{array} \quad G_1 = \begin{array}{c}
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\end{array} \)

(simplified protocol)
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 = \begin{array}{c}
\text{ vertices } \\
\hline
\end{array}$, $G_1 = \begin{array}{c}
\text{ vertices } \\
\hline
\end{array}$

$H = \pi(G_b)$, $b \leftarrow \{0,1\}$, $\pi \leftarrow S_n$.

(simplified protocol)
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: \( G_0 = \) \( G_1 = \)

\[ H = \pi(G_b) \quad b \leftarrow \{0,1\}, \pi \leftarrow S_n. \]

(simplified protocol)
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance:  $G_0 = \begin{array}{c}
\text{Node 1} \\
\text{Node 2} \\
\text{Node 3}
\end{array}$  \quad  $G_1 = \begin{array}{c}
\text{Node 1} \\
\text{Node 2} \\
\text{Node 3}
\end{array}$

\[ H = \pi(G_b) \]

\[ b' \]

(simplified protocol)

\[ b \leftarrow \{0,1\}, \pi \leftarrow S_n. \]

\[ V \text{ accepts if } b' = b. \]

Repeat for soundness.
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: \( G_0 = \) ![Graph 1] \( G_1 = \) ![Graph 2]

\[
H = \pi(G_b) \quad b \leftarrow \{0,1\}, \pi \leftarrow S_n.
\]

V accepts if \( b' = b \).
Repeat for soundness.

• \( P \) can distinguish \( \pi(G_0), \pi(G_1) \) iff graphs are not isomorphic.
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: \( G_0 \)

\[
H = \pi(G_b)
\]

\( b \leftarrow \{0,1\}, \pi \leftarrow S_n. \)

\( V \) accepts if \( b' = b. \)

Repeat for soundness.

- \( P \) can distinguish \( \pi(G_0), \pi(G_1) \) iff graphs are not isomorphic.
- ZK holds against honest verifiers, since honest verifier already knows \( b. \)
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

 Instance: \( G_0 = \) \hspace{1cm} \( G_1 = \)

\[ H = \pi(G_b) \]

\[ b \leftarrow \{0,1\}, \pi \leftarrow S_n. \]

\( V \) accepts if \( b' = b \).

Repeat for soundness.

- \( P \) can distinguish \( \pi(G_0), \pi(G_1) \) iff graphs are not isomorphic.
- ZK holds against honest verifiers, since honest verifier already knows \( b \).

In the full [GMW86] protocol, \( V \) proves that it knows \( b \) before \( P \) sends \( b' \).
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: \( G_0 = \) \( G_1 = \)

\[ H = \pi(G_b) \]

\( b \leftarrow \{0,1\}, \pi \leftarrow S_n. \)

[GMW86] Proof-of-Knowledge Subprotocol: \( V \) proves it “knows” \( b. \)
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

**Instance:** \( G_0 = \) ![Graph 1] \( G_1 = \) ![Graph 2]

\( H = \pi(G_b) \)

\( b \leftarrow \{0,1\}, \pi \leftarrow S_n. \)

[GMW86] Proof-of-Knowledge Subprotocol: \( V \) proves it “knows” \( b. \)

For \( i \in [m] \):

\( G_{i,0}, G_{i,1} \) randomly permute \((G_0, G_1) \) or \((G_1, G_0)\)
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance:  \( G_0 = \begin{array}{c}
\text{Instance 1} \\
\end{array} \quad G_1 = \begin{array}{c}
\text{Instance 2}
\end{array} \)

\[ H = \pi(G_b) \]

\[ b \leftarrow \{0,1\}, \pi \leftarrow S_n. \]

[GMW86] Proof-of-Knowledge Subprotocol: \( V \) proves it “knows” \( b \).

For \( i \in [m] \):

\[
G_{i,0}, G_{i,1} \quad \text{randomly permute} \ (G_0, G_1) \text{ or } (G_1, G_0)
\]
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: \( G_0 = \) [Diagram of graph 0] \( G_1 = \) [Diagram of graph 1]

\[ H = \pi(G_b) \]

\[ b \leftarrow \{0,1\}, \pi \leftarrow S_n. \]

[GMW86] Proof-of-Knowledge Subprotocol: \( V \) proves it “knows” \( b \).

For \( i \in [m] \):

\[ r_i \leftarrow \{0,1\} \]

randomly permute \((G_0, G_1)\) or \((G_1, G_0)\)
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: \(G_0 = \begin{graph}
\node[draw] at (0,0) {};
\node[draw] at (1,0) {};
\node[draw] at (2,0) {};
\node[draw] at (1,1) {};
\draw (0,0) -- (1,1) -- (2,0);
\end{graph}\) \(G_1 = \begin{graph}
\node[draw] at (0,0) {};
\node[draw] at (1,0) {};
\node[draw] at (2,0) {};
\node[draw] at (1,1) {};
\draw (0,0) -- (1,1) -- (2,0);
\end{graph}\)

\[
H = \pi(G_b)
\]

\(b \leftarrow \{0,1\}, \pi \leftarrow S_n.\)

[GMW86] Proof-of-Knowledge Subprotocol: \(V\) proves it "knows" \(b\).

For \(i \in [m]\): \(G_{i,0}, G_{i,1}\)

\(r_i \leftarrow \{0,1\}\)

\(r_i\)

\(z_i\)

randomly permute \((G_0, G_1)\) or \((G_1, G_0)\)

If \(r_i = 0\), reveal both permutations

If \(r_i = 1\), reveal permutation between \(H\) and one of \(G_{i,0}, G_{i,1}\)
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: \( G_0 = \) \hspace{1cm} \( G_1 = \)

\[ H = \pi(G_b) \]
\[ b \leftarrow \{0,1\}, \pi \leftarrow S_n. \]

[GMW86] Proof-of-Knowledge Subprotocol: \( V \) proves it “knows” \( b \).

For \( i \in [m] \):

\[ r_i \leftarrow \{0,1\} \]

\[ G_{i,0}, G_{i,1} \]

randomly permute \((G_0, G_1)\) or \((G_1, G_0)\)

\[ z_i \]

If \( r_i = 0 \), reveal both permutations

If \( r_i = 1 \), reveal permutation between \( H \) and one of \( G_{i,0}, G_{i,1} \)

Why does this prove knowledge of \( b \)? If \( V^* \) can answer both challenges correctly, it must know which of \( G_0, G_1 \) is isomorphic to \( H \)
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 = \square$, $G_1 = \bigtriangleup$

$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

$H = \pi(G_b)$


For $i \in [m]$:

$r_i \leftarrow \{0,1\}$

randomly permute $(G_0,G_1)$ or $(G_1,G_0)$

If $r_i = 0$, reveal both permutations

If $r_i = 1$, reveal permutation between $H$ and one of $G_{i,0}, G_{i,1}$

$b'$
Extract-and-Simulate Paradigm

ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$. 
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator
1) Run $V^*$ once, check if $z$ is accepting. If not, done.

Instance:

\[ G_0, G_1 \]
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator
1) Run $V^*$ once, check if $z$ is accepting. If not, done.
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator

1) Run $V^*$ once, check if $z$ is accepting. If not, done.

Instance:

- $G_0, G_1$
- $H, \{G_{i,0}, G_{i,1}\}_i$

Isomorphic to $G_b$ (supposedly)
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

**[GMW86] ZK Simulator**

1) Run $V^*$ once, check if $z$ is accepting. If not, done.

Instance: $G_0, G_1$

isomorphic to $G_b$ (supposedly)

\[ H, \{G_{i,0}, G_{i,1}\}_i \]

\[ r \]

\[ z \]
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator

1) Run $V^*$ once, check if $z$ is accepting. If not, done.
2) Rewind $V^*$ and query it until it outputs a second accepting $z'$.

Instance: $G_0, G_1$

isomorphic to $G_b$ (supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind

$r$

$z$

$V^*$
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

**[GMW86] ZK Simulator**

1) Run $V^*$ once, check if $z$ is accepting. If not, done.
2) Rewind $V^*$ and query it until it outputs a second accepting $z'$. 

Instance:

$G_0, G_1$

isomorphic to $G_b$ (supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator

1) Run $V^*$ once, check if $z$ is accepting. If not, done.
2) Rewind $V^*$ and query it until it outputs a second accepting $z'$.
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator
1) Run $V^*$ once, check if $z$ is accepting. If not, done.
2) Rewind $V^*$ and query it until it outputs a second accepting $z'$.
Extract-and-Simulate Paradigm

ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator

1) Run $V^*$ once, check if $z$ is accepting. If not, done.
2) Rewind $V^*$ and query it until it outputs a second accepting $z'$.
3) Given two accepting transcripts $(r, z), (r', z')$ where $r \neq r'$, simulator can extract $b$.

Instance: $G_0, G_1$ isomorphic to $G_b$ (supposedly)

Rewind $V^*$

$H, \{G_{i,0}, G_{i,1}\}_i$

$r'$

$z'$
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator

1) Run $V^*$ once, check if $z$ is accepting. If not, done.
2) Rewind $V^*$ and query it until it outputs a second accepting $z'$.
3) Given two accepting transcripts $(r, z), (r', z')$ where $r \neq r'$, simulator can extract $b$.

Instance: $G_0, G_1$ isomorphic to $G_b$

rewind

$z'$$r'$ $V^*$

$b$
Extract-and-Simulate Paradigm

ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator
1) Run $V^*$ once, check if $z$ is accepting. If not, done.
2) Rewind $V^*$ and query it until it outputs a second accepting $z'$.
3) Given two accepting transcripts $(r, z), (r', z')$ where $r \neq r'$, simulator can extract $b$.

This “extract and simulate” approach appears in many textbook ZK protocols [GMR85,GMW86,FS90]
ZK requires a simulator that outputs $b$ if $V^*$ outputs accepting $z$.

[GMW86] ZK Simulator
1) Run $V^*$ once, check if $z$ is accepting. If not, done.
2) Rewind $V^*$ and query it until it outputs a second accepting $z'$.
3) Given two accepting transcripts $(r, z), (r', z')$ where $r \neq r'$, simulator can extract $b$.

Important point: what is the simulator’s runtime?
Extract-and-Simulate Paradigm

ZK requires a simulator that outputs \( b \) if \( V^* \) outputs accepting \( z \).

[GMW86] ZK Simulator
1) Run \( V^* \) once, check if \( z \) is accepting. If not, done.
2) Rewind \( V^* \) and query it until it outputs a second accepting \( z' \).
3) Given two accepting transcripts \((r, z), (r', z')\) where \( r \neq r'\), simulator can extract \( b \).

Instance:
\[
G_0, G_1
\]

isomorphic to \( G_b \) (supposedly)

Important point: what is the simulator’s runtime?

ZK def allows \textit{expected poly-time} simulation \((\varepsilon \cdot \frac{1}{\varepsilon} = 1)\)
Plan for Today

1. Recap: GNI protocol + “extract-and-simulate” ✓

2. Challenges in the post-quantum setting

3. This work: defining post-quantum ZK

4. This work: quantum extract-and-simulate
For post-quantum ZK, we need to:

(1) Extract $b$ from quantum $V^*$. 
For post-quantum ZK, we need to:

(1) Extract $b$ from quantum $V^*$.  

- Classically sim records $z$ and rewinds $V^*$ to an earlier state.
For post-quantum ZK, we need to:

(1) Extract $b$ from quantum $V^*$.
- Classically sim records $z$ and rewinds $V^*$ to an earlier state.
- If $V^*$ is quantum, recording $z$ can disturb $V^*$’s state!
For post-quantum ZK, we need to:

1. **Extract \( b \) from quantum \( V^* \).**
   - Classically sim records \( z \) and rewinds \( V^* \) to an earlier state.
   - If \( V^* \) is quantum, recording \( z \) can disturb \( V^* \)’s state!

2. **Simulate internal quantum state of \( V^* \).**
For post-quantum ZK, we need to:

(1) Extract $b$ from quantum $V^\ast$.
   - Classically sim records $z$ and rewinds $V^\ast$ to an earlier state.
     
     ![Diagram](image)
     
     - If $V^\ast$ is quantum, recording $z$ can disturb $V^\ast$’s state!

(2) Simulate internal quantum state of $V^\ast$.
   - \[U12, CMSZ21\] can handle (1), but will disturb the state.
For post-quantum ZK, we need to:

(1) Extract $b$ from quantum $V^*$. 
• Classically sim records $z$ and rewinds $V^*$ to an earlier state.

(2) Simulate internal quantum state of $V^*$. 
• $[U12, CMSZ21]$ can handle (1), but will disturb the state.

(3) Run in expected quantum poly time.
For post-quantum ZK, we need to:

(1) Extract $b$ from quantum $V^*$.  
- Classically sim records $z$ and rewinds $V^*$ to an earlier state.
- If $V^*$ is quantum, recording $z$ can disturb $V^*$'s state!

(2) Simulate internal quantum state of $V^*$.  
- $[U12, CMSZ21]$ can handle (1), but will disturb the state.

(3) Run in expected quantum poly time.  
- Defining variable-time quantum computation is subtle. $[M97, O98, LP98]$
Plan for Today

1. Recap: GNI protocol + “extract-and-simulate” ✓
2. Challenges in the post-quantum setting ✓
3. This work: defining post-quantum ZK
4. This work: quantum extract-and-simulate
How should post-quantum ZK even be defined?
(for expected poly-time simulation)
How should post-quantum ZK even be defined?
(for expected poly-time simulation)

Original ZK def [GMR85]:
View of any *classical* poly-time $V^*$ can be simulated in *classical* *expected* poly time.*
How should post-quantum ZK even be defined?
(for expected poly-time simulation)

Original ZK def [GMR85]:
View of any classical poly-time $V^*$
can be simulated in classical expected poly time.*

[BL02]: Classical strict poly-time ZK sim is impossible (black-box, constant-round).
How should post-quantum ZK even be defined? (for expected poly-time simulation)

### Original ZK def [GMR85]:
View of any **classical** poly-time $V^*$ can be simulated in **classical** expected poly time. *

### [BL02]:
Classical **strict** poly-time ZK sim is impossible (black-box, constant-round).

### Obvious (?) post-quantum ZK def
View of any **quantum** poly-time $V^*$ can be simulated in **quantum** expected poly time.
How should post-quantum ZK even be defined? 
(for expected poly-time simulation)

Original ZK def [GMR85]:
View of any \textit{classical} poly-time $V^*$ can be simulated in \textit{classical} \textit{expected} poly time.*

[BL02]: Classical \textit{strict} poly-time ZK sim is impossible (black-box, constant-round).

Obvious (?) post-quantum ZK def
View of any \textit{quantum} poly-time $V^*$ can be simulated in \textit{quantum} \textit{expected} poly time.

[CCLY21]: Quantum \textit{expected} poly-time ZK sim is impossible* (black-box, constant-round).
How should post-quantum ZK even be defined? (for expected poly-time simulation)

Original ZK def [GMR85]:
View of any classical poly-time $V^*$ can be simulated in classical expected poly time.*

[BL02]: Classical strict poly-time ZK sim is impossible (black-box, constant-round).

Obvious (?) post-quantum ZK def
View of any quantum poly-time $V^*$ can be simulated in quantum expected poly time.

[CCLY21]: Quantum expected poly-time ZK sim is impossible* (black-box, constant-round).

This work: new ZK def that circumvents CCLY21 barrier.
This Work: Measured vs. Coherent EQPT Simulation
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

halt qubit $|0\rangle$ —
input $|\psi\rangle$ —
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

- halt qubit $|0\rangle$
- input $|\psi\rangle$

Diagram:

- Step 1
- $M$
- $0$
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

halt qubit $\ket{0}$
input $\ket{\psi}$
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

-halt qubit $|0\rangle$
- input $|\psi\rangle$

$M$

step 1

0

$M$

step 2

0

$\ldots$

$M$

step $t$

1

$|\psi_t\rangle$

$\mathbb{E}[t] = \text{poly}(\lambda)$
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

\[ |0\rangle \quad \text{step 1} \quad 0 \quad \text{step 2} \quad 0 \quad \text{step } t \quad 1 \quad |\psi_t\rangle \]

\[ \mathbb{E}[t] = \text{poly}(\lambda) \]

[CCLY21]: measuring simulator’s runtime disturbs verifier’s state.
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

Coherent EQPT

\[ U \text{ runs a measured-EQPT procedure coherently for } 2^\lambda \text{ steps, leaving } t \text{ in superposition.} \]
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

Coherent EQPT

\[ U \text{ runs a measured-EQPT procedure coherently for } 2^\lambda \text{ steps, leaving } t \text{ in superposition.} \]
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

| halt qubit | input $|\psi\rangle$ |
|------------|----------------|
| $|0\rangle$ | $M$ |
| $M$ | $|\phi\rangle$ |
| $|\psi_t\rangle$ |

$\mathbb{E}[t] = \text{poly}(\lambda)$

[CCLY21]: measuring simulator’s runtime disturbs verifier’s state.

Coherent EQPT

| runtime $|0^\lambda\rangle$ | input $|\psi\rangle$ |
|-----------------|----------------|
| $U$ | $= \sum_t \alpha_t |t\rangle|\psi_t\rangle$ |

If we measure $t$ here,

$\mathbb{E}[t] = \text{poly}(\lambda)$

$U$ runs a measured-EQPT procedure coherently for $2^\lambda$ steps, leaving $t$ in superposition.
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

\[ |0\rangle \quad M \quad |0\rangle \quad M \quad ... \quad M \quad |\psi_t\rangle \]

\[ E[t] = \text{poly}(\lambda) \]

[CCLY21]: measuring simulator’s runtime disturbs verifier’s state.

Coherent EQPT

\[ |0^\lambda\rangle \quad U \quad |\psi\rangle \quad C \]

\[ = \sum_t \alpha_t |t\rangle |\psi_t\rangle \]

\( U \) runs a measured-EQPT procedure coherently for \( 2^\lambda \) steps, leaving \( t \) in superposition. \( C \) is a unitary (e.g., SWAP).
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

- **Coherent EQPT**
  - $U$ runs a measured-EQPT procedure coherently for $2^\lambda$ steps, leaving $t$ in superposition. $C$ is a unitary (e.g., SWAP).

[CCLY21]: measuring simulator’s runtime disturbs verifier’s state.
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

- $|0\rangle$ is the halt qubit.
- $|\psi\rangle$ is the input.
- $M$ is the measurement.

$$\mathbb{E}[t] = \text{poly}(\lambda)$$

[CCLY21]: measuring simulator’s runtime disturbs verifier’s state.

Coherent EQPT

- $|0^\lambda\rangle$ is the runtime.
- $|\psi\rangle$ is the input.
- $C$ is a unitary (e.g., SWAP).
- $U$ runs a measured-EQPT procedure coherently for $2^\lambda$ steps, leaving $t$ in superposition.

$U$ runs a measured-EQPT procedure coherently for $2^\lambda$ steps, leaving $t$ in superposition. $C$ is a unitary (e.g., SWAP).
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

\[ \text{hal} \quad \begin{array}{c} |0\rangle \\
\text{inpu} \\
|\psi\rangle \\
\end{array} \longrightarrow \begin{array}{c} M \\
|\text{step} 1\rangle \\
0 \\
\end{array} \longrightarrow \begin{array}{c} M \\
|\text{step} 2\rangle \\
0 \\
\end{array} \longrightarrow \begin{array}{c} \ldots \\
M \\
|\text{step } t\rangle \\
1 \\
\end{array} \longrightarrow |\psi_t\rangle \]

\[ \mathbb{E}[t] = \text{poly}(\lambda) \]

[CCLY21]: measuring simulator’s runtime disturbs verifier’s state.

Coherent EQPT

\[ \begin{array}{c} \text{run} \\
|0^\lambda\rangle \\
\text{inpu} \\
|\psi\rangle \\
\end{array} \longrightarrow \begin{array}{c} U \\
|0^\lambda\rangle \\
\mathcal{C} \\
\end{array} \longrightarrow \begin{array}{c} U^\dagger \\
|\psi_t\rangle \\
\end{array} \]

\[ = \sum_t \alpha_t |t\rangle |\psi_t\rangle \]

\( U \) runs a measured-EQPT procedure coherently for \( 2^\lambda \) steps, leaving \( t \) in superposition. \( \mathcal{C} \) is a unitary (e.g., SWAP).

Coherent EQPT leaves runtime in superposition and later uncomputes it.

CCLY21 does not rule out coherent EQPT simulation!
Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?
Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

Related: why is EPT simulation reasonable in classical ZK?
Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

**Related:** why is EPT simulation reasonable in classical ZK?

1) EPT is no stronger than poly-time.
Ex: if assumption A is broken in EPT, it’s also broken in poly time.
Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

**Related:** why is EPT simulation reasonable in classical ZK?

1) EPT is no stronger than poly-time.
Ex: if assumption A is broken in EPT, it’s also broken in poly time.

2) EPT appears to be the weakest model that makes ZK possible.*
[BL02]: poly-time sim is impossible.*

*for constant-round protocols w/ black-box simulation
Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

We show an analogous state of affairs for coherent EQPT.
Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

We show an analogous state of affairs for coherent EQPT.

1) Coherent EQPT is no stronger than QPT.
Ex: if assumption A is broken in coherent EQPT, it’s also broken in QPT.
Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

We show an analogous state of affairs for coherent EQPT.

1) Coherent EQPT is no stronger than QPT. 
   Ex: if assumption A is broken in coherent EQPT, it’s also broken in QPT.

2) Coherent EQPT appears to be the weakest model that makes post-quantum ZK possible.*
   [CCLY21]: measured EQPT sim impossible.*

*for constant-round protocols w/ black-box simulation
Plan for Today

1. Recap: GNI protocol + “extract-and-simulate” ✓
2. Challenges in the post-quantum setting ✓
3. This work: defining post-quantum ZK ✓
4. This work: quantum extract-and-simulate
Goal: build coherent EQPT simulator that extracts $b$ from $V^*$ without disturbing its internal state $|\psi\rangle$. 
[CMSZ21] allows us to extract $b$, but will disturb $|\psi\rangle$. 
[CMSZ21] allows us to extract $b$, but will disturb $|\psi\rangle$.

CMSZ rewinding: define projector $\Pi$ onto “useful” verifier states.
[CMSZ21] allows us to extract \( b \), but will disturb \(|\psi\rangle\).

CMSZ rewinding: define projector \( \Pi \) onto “useful” verifier states.
[CMSZ21] allows us to extract $b$, but will disturb $|\psi\rangle$.

CMSZ rewinding: define projector $\Pi$ onto "useful" verifier states.

\[ V^* \]

\[ |\psi\rangle \]

\[ z_1 \]

\[ |\psi'\rangle \]

\[ \text{Amplify onto } \Pi \]

\[ r_1 \]

\[ V^* \]

\[ |\psi_1\rangle \]
[CMSZ21] allows us to extract $b$, but will disturb $|\psi\rangle$.

CMSZ rewinding: define projector $\Pi$ onto “useful” verifier states.
[CMSZ21] allows us to extract $b$, but will disturb $|\psi\rangle$. 

CMSZ rewinding: define projector $\Pi$ onto “useful” verifier states.

Idea: define variable-time procedure

$\text{Ext} =$ “Run CMSZ to get valid $(r, z), (r', z')$”
[CMSZ21] allows us to extract $b$, but will disturb $|\psi\rangle$.

CMSZ rewinding: define projector $\Pi$ onto “useful” verifier states.

\[
\begin{align*}
V^* & \quad |\psi\rangle \quad r_1 \quad z_1 \quad |\psi\rangle \\
& \quad \downarrow \\
& \quad |\psi\rangle \quad \downarrow \quad \text{Amplify onto } \Pi \\
& \quad \downarrow \\
V^* & \quad |\psi_1\rangle \quad r_2 \quad z_2 \quad |\psi_1\rangle \\
& \quad \downarrow \\
& \quad \quad \vdots
\end{align*}
\]

**Idea:** define variable-time procedure

\[\text{Ext} = \text{“Run CMSZ to get valid } (r, z), (r', z') \text{”} \]

**Simulator:**

1) Run $U_{\text{Ext}}$ (coherent version of Ext)
2) Extract $b$ from $(r, z), (r', z')$
3) Run $U_{\text{Ext}}^*$
[CMSZ21] allows us to extract $b$, but will disturb $|\psi\rangle$.

CMSZ rewinding: define projector $\Pi$ onto “useful” verifier states.

Idea: define variable-time procedure

$$\text{Ext} = \text{“Run CMSZ to get valid } (r, z), (r', z') \text{”}$$

Simulator:
1) Run $U_{\text{Ext}}$ (coherent version of $\text{Ext}$)
2) Extract $b$ from $(r, z), (r', z')$
3) Run $U_{\text{Ext}}^\dagger$

This works if:
- $\text{Ext}$ is measured-EQPT
- $\text{Ext}$ succeeds with probability $\approx 1$
- $b$ is unique.
Big problem: CMSZ-based Ext doesn’t run in expected poly-time!
Big problem: CMSZ-based Ext doesn’t run in expected poly-time!

If $V^*$ gives valid $z$ with prob $\varepsilon$, it takes $1/\varepsilon$ expected loops of “measure + repair” to get another accepting response.
Big problem: CMSZ-based Ext doesn’t run in expected poly-time!

If $V^*$ gives valid $z$ with prob $\varepsilon$, it takes $1/\varepsilon$ expected loops of “measure + repair” to get another accepting response.

Expected runtime is $\varepsilon \left( \frac{1}{\varepsilon} \right) \left( \frac{1}{\varepsilon^4} \right) = \text{unbounded}$

Classical runtime | CMSZ repair runtime
**Big problem:** CMSZ-based Ext doesn’t run in expected poly-time!

If $V^*$ gives valid $z$ with prob $\varepsilon$, it takes $1/\varepsilon$ expected loops of “measure + repair” to get another accepting response.

Expected runtime is $\varepsilon \left( \frac{1}{\varepsilon} \right) \left( \frac{1}{\varepsilon^4} \right) = \text{unbounded}$

**This work:**

- New rewinding template where accepting $(r, z)$ is generated by amplifying adversary onto *guaranteed accepting executions* instead of querying on random $r$. 
Big problem: CMSZ-based Ext doesn’t run in expected poly-time!

If $V^*$ gives valid $z$ with prob $\varepsilon$, it takes $1/\varepsilon$ expected loops of “measure + repair” to get another accepting response.

Expected runtime is $\varepsilon \left( \frac{1}{\varepsilon} \right) \left( \frac{1}{\varepsilon^4} \right) = \text{unbounded}$

This work:

• New rewinding template where accepting $(r, z)$ is generated by amplifying adversary onto guaranteed accepting executions instead of querying on random $r$.

• Not obvious: why does repair still work?
Big problem: CMSZ-based \texttt{Ext} doesn’t run in expected poly-time!

If $V^\star$ gives valid $z$ with prob $\varepsilon$, it takes $1/\varepsilon$ expected loops of “measure + repair” to get another accepting response.

\[
\text{Expected runtime is } \varepsilon \left(\frac{1}{\varepsilon}\right) \left(\frac{1}{\varepsilon^4}\right) = \text{unbounded}
\]

This work:
- New rewinding template where accepting $(r, z)$ is generated by amplifying adversary onto \textit{guaranteed accepting executions} instead of querying on random $r$.
- Not obvious: why does repair still work?
- Further speed up CMSZ-style repair with new, variable-time algorithms based on the quantum singular value transform [GSLW19].
Conclusions

• Definitions and reductions for post-quantum ZK are subtle!
• We define post-quantum ZK using coherent EQPT simulation.
• We build coherent EQPT simulators for textbook ZK protocols by combining a new rewinding template with variable-time algorithms based on [CMSZ21,GLSW19].
Conclusions

• Definitions and reductions for post-quantum ZK are subtle!
• We define post-quantum ZK using coherent EQPT simulation.
• We build coherent EQPT simulators for textbook ZK protocols by combining a new rewinding template with variable-time algorithms based on [CMSZ21, GLSW19].

Future directions

• Develop general theory of post-quantum ZK.
• Quantum-communication ZK? Natural starting point: [BG20, GJMZ22]

Thanks for listening!